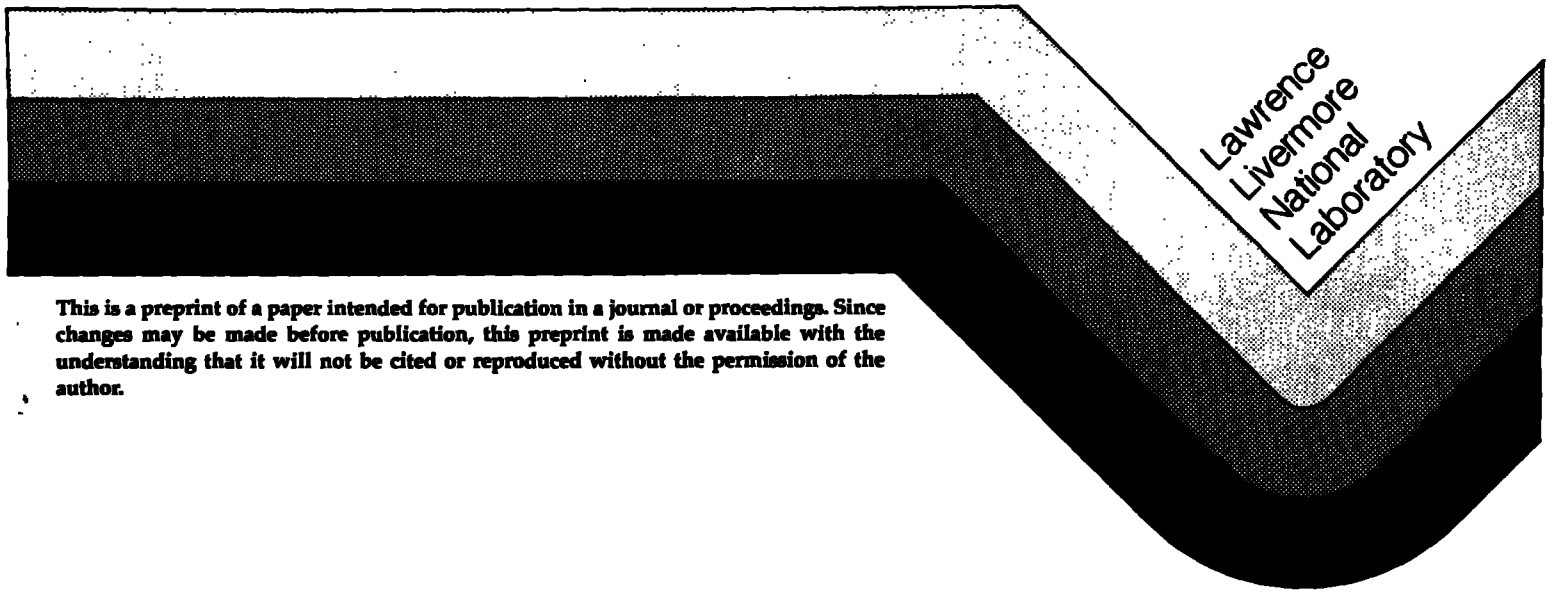
UCRL- 93404
PREPRINT

Fusing AI and Simulation in Military Modeling

Stanley A. Erickson, Jr.

Artificial Intelligence in Simulation Conference
Ghent, Belgium
February 25-28, 1985

September 1985



FUSING AI AND SIMULATION IN MILITARY MODELING*

Stanley A. Erickson, Jr.
University of California
Lawrence Livermore National Laboratory
P. O. Box 808
Livermore, California 94550

ABSTRACT

Military simulation can be significantly improved through use of Artificial Intelligence to simulate tactical decision-making. Our project has served to test out many tentative ideas on how to accomplish this, including unique architectures for both hardware and software, methods of eliciting clear rules from multiple topic area experts in a chain of command, how to make programming more effective, and on ways of writing the AI modules. This paper serves as an interim status report.

INTRODUCTION

The objective of this paper is to describe the AI/Simulation project underway for the last three years at Lawrence Livermore National Laboratory. We wish to discuss the background of the project, the four phases of work: problem definition, conceptualization of the solution, architectural design and the evolving implementation of the testbed simulation, and finally what was learned or concluded from each phase. By necessity, things learned in the latter stage, where concepts are put to the test, influenced the earlier conclusions of the first three phases, and a complete partition is not possible. Nevertheless, separating out these phases provides a convenient framework to discuss the entirety of the project.

The work belongs to the genre of military simulation, which is possibly more prevalent than any other type of simulation, but less reported on, especially outside of contractor's reports, which are often classified. Some theoretical work in military simulation has been done, but the majority of the efforts in this field have been devoted to solving specific problems of military science and engineering design. Excellent broad references to the wide variety of military simulation are the two proceedings of symposia by Huber [1,2], and the compendium of Hughes [3].

Fusing AI into military simulation is one of the frontier areas of military simulation research. Over the last 25 years, paralleling the advances in computational capability, military simulation has taken on more and more aspects of reality. The initial simulations of the early 1960's involved the flight of a projectile or missile, and its closure on a target. Later, multiple objects were included, with the emphasis on the maneuvering of one vehicle in response to another's motions. At the same time, sensor systems, such as radar and sonar, were modeled, an effort which involved the use of signal propagation and detection models, signal processing, tracking algorithms and the classification of targets according to what electromagnetic or acoustic signatures they presented. In the 1970's, a great deal of progress was made in modeling communications, and in the latter part of the decade, in electronic warfare, principally jamming and counterdetection of signals by unintended recipients. Most recently, interest has turned to tactics and strategy.

The first phase of the work was the problem definition: specifically how can one overcome the programming difficulty and other inadequate methods of representing tactical decisions. In order to understand the problem, it is useful to first appreciate pre-AI methods of representing decision-making, and the features of military simulation which generate the complexity causing these difficulties.

PRE-AI METHODS OF REPRESENTING DECISIONS IN SIMULATION

Strategy and tactics have traditionally been incorporated in simulations by using decision trees or decision matrices, or in the case of wargames or interactive simulations, by utilizing human actors to make decisions. Only a minimal set of options can be incorporated with decision trees or variations of them, and there is a question with their use as to the adequacy of the decisions.

Decision trees attempt to model the time flow of decisions and events in a simulation. As shown in Figure 1, a decision tree consists of a set of alternating decisions and simulation events. An initial event can start off the simulation, and then one or both sides in the simulation make a decision as to what response to make. Then the simulation moves forward, incorporating the actions which result from these decisions, until another significant event occurs. At this point, the sides in the simulation make another decision. Because the event which occurs is not immediately predictable, either because of the complexity of the situation or because of random variables in the simulation, the designer of a decision-tree based simulation must predict the complete set of possible options and decisions at each stage. For a lengthy simulation, this

*This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

represents a tremendous burden. Typical examples of a decision tree type of problem are games with random events, such as poker or gin rummy. Poker has a short enough series of moves and random events as to make it possible to write programs to play the game. Gin rummy, and military simulations, simply have too many moves to be realistically programmable.

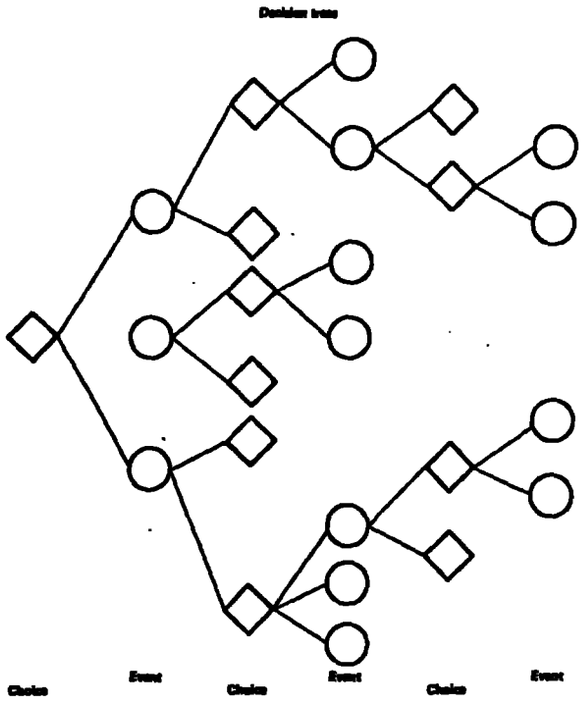


Figure 1: The decision tree methodology for modeling decisions in simulations consists of collecting and organizing in temporal sequence all possible major events and decisions, and then creating rules as to what decision options are to be chosen after what events occur.

Tricks used to write simulations using the decision tree paradigm in complicated simulations relate to the use of multiple decision trees, corresponding to different phases of the simulation. Sometimes, with only a small number of objects involved, a simulation can be broken up into a search phase, a closing phase, a weapon launching phase, as examples. However, with many objects, or with long phases such as would be caused by objects capable of sophisticated actions, such as maneuvering fighter aircraft, even this device soon fails. Other modifications include re-entrant trees in which the monotonicity of time is abandoned in the decision tree, and the chain of events is allowed to close back on itself at any arbitrary point. In this type of decision tree simulation, the problem that arises is defining which situations are identical and which are sufficiently distinct as to require a wholly new branch of the tree. In simulations with literally hundreds of variables, making this identification becomes overwhelmingly difficult.

Decision matrices, on the other hand, deal with snapshots of the simulation. If history of the simulation is of no importance, and decisions can be made solely on the basis of a reasonable number of variables in the simulation, then those variables can be translated into decisions for each side. As illustrated in Figure 2, the variables which enter into a decision matrix are typically binary. Continuous or multi-valued discrete variables are turned into binary variables by using thresholds: if x, a simulation variable such as altitude, is greater than x_0, then X, a decision tree variable, is + (or true), otherwise it is -.

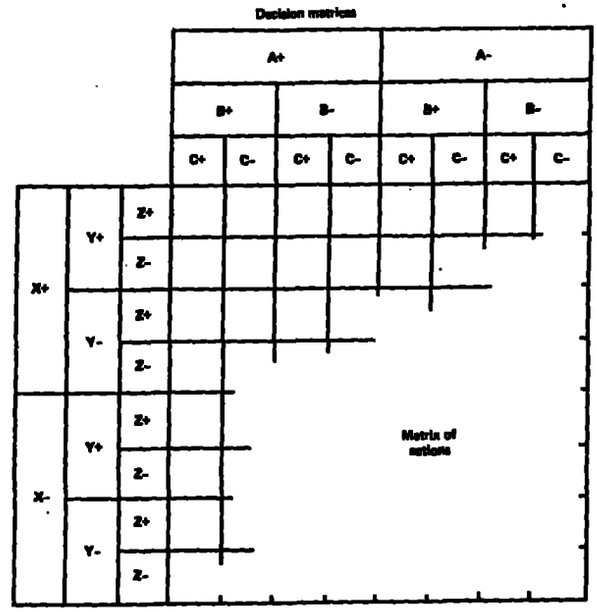


Figure 2: The decision matrix methodology for modeling decisions consists of describing the state of the simulation in terms of a finite number of variables, and then of creating a relationship between the instantaneous state of the simulation with the chosen decision.

One difficulty which arises with decision matrices is the number that are required, and the number of variables which are included in each. Too many variables in each situation make it difficult to create the combinatorially large number of options. One method used to alleviate the combinatorics is to make hierarchies of variables, in which lower levels of variables are only used in the evaluation if higher levels of variables take on certain subsets of possible values. However, this only reduces the number of combinations by a order of magnitude in typical situations.

Sometimes there is a major imbalance between the representation of decisions on the two sides of the conflict. Typically the modeler's own side is better represented, as he knows or can learn the details of his own side's tactics, and the details of his own side's vehicle, weapon, sensor and communication systems can be incorporated. When this is available, the own-side's tactics can be adjusted to take care of the details of these systems, making their use closer to optimal. Tactics and system-tactics interactions of the other side are less worked on, and less well represented. This is often referred to as the "dumb enemy assumption," and involves the enemy forces using simple attack strategies, not maneuvering to their best advantage, not allocating their firepower well, or sometimes just serving as targets.

This situation leads to poor system choices, for in most military situations, tactics strongly influence the outcome of a conflict. Better equipment does not necessarily imply better results.

The use of human actors to play roles in a simulation has some advantages but also a number of serious drawbacks. Chance is an important factor in conflict situations, and to understand its influence it is good to have a statistical number of replications of the simulation. With human actors, each replication differs, as the actors learn from the past ones, and modify their behavior. They almost never stick to the same tactics. They get bored. Furthermore, human actors impose running time limitations on the simulation. It cannot run too many times faster than real time in order to allow the simulation to seem real to the actors, who are often military officers playing the game to utilize their experience.

Human actors also need a minimum amount of time to make decisions, and this puts a restriction on the number of plays that can be accomplished in a day. If one wants to do a sensitivity study to determine the best value for one or many parameters, it is simply not feasible to use human actors and conclude anything verifiable. Simulation with human players is "insightful," rather than being a quantitative tool to design systems. It has many uses, especially in training, but analysis is not one of them.

The project that is discussed here was initially created to experiment with a combination of Artificial Intelligence, used to model human decision-making, and military simulation, used to model the physical world. Replacing a war-game-playing expert with a war-game-playing expert system would alleviate the problems found both in all-computer simulations and in war games. Expert systems are designed to deal with combinatorially complex situations, and hopefully will provide much more realistic decision-making than decision trees or decision matrices. Furthermore, they may be able to run at several times real time, allowing statistical numbers of replications to take place with various values for parameter sets in reasonable amounts of computer time. However, as work has continued, the division of the simulation code between AI and non-AI has grown blurry, for several reasons. But first, a discussion of the topic of the simulation is in order.

MILITARY SIMULATION FEATURES

There is no disputing the inherent complexity of military operations. Whoever coined the phrase, "the fog of war," was speaking of decision-making in a very uncertain, sometimes data-poor environment, with harsh time constraints, and limited replanning capability. There are typically a large number of independent decision-making entities, engaged both in taking actions, and communicating to superiors and subordinates. The information handling problem alone is immense, with information being combined and compressed up the chain of command, and plans and orders being elaborated down the chain of command. Communication is not necessarily correct, adequate, or timely, or even possible. Information on the situation may be incorrect, owing to a limit on the operation of information gathering (intelligence) units.

Besides these problems, referred to collectively as C^3I, for command, control, communication and intelligence, there are difficulties concerned with the actual environment in which the operation takes place. Sensing of the environment or of the vehicles and vessels (platforms) involved may be quite imperfect owing to the distance between sensor and object, due to obscuration either natural or man-made, due to camouflage or deception, due to intermittency of data collection, due to a limitation on the bandwidth of the sensor, or due to jamming. The recipient of the information is faced with a more difficult than usual interpretation problem.

There are two general tasks for a decision maker to perform, occurring at lower and higher command levels. At a lower command level, tactics is the name of the game. The operator of a platform, or a sensor, attempts to use it to accomplish some goal or goals, often with multiple constraints. The operator of a platform, say a aircrew or tankcrew, has a ruleset which has actions for the platform as the outputs. The operator of a sensor, say a radar or a sonar, may have some actions to be taken, such as aiming or frequency choice, but the primary output is the sending of messages based on received information. One impressive state of the art simulation at this command level is TAC Warrior, in which multiple maneuvering aircraft are simulated with all the details of their motion and pilot control functions.

At the higher command levels, information alone is transmitted in and out via message traffic. The job of the command level is to decide what resources will be applied to what mission; this is done with the expectation of certain results. Each of these higher command levels has a mission or set of missions, and attempts to satisfy them in priority order. For each typical situation, there are procedures to be followed, which have been worked out in advance based on practice and experience. Atypical situations have to be worked out in real time based on more primitive expectation models.

Thus there are four generic tasks to simulate with AI. One is the operation of a platform, another is the compression of sensor information. There also is a procedure following behavior for allocation of resources, and finally the most difficult, planning in difficult situations.

After the problem definition effort had matured, it was possible to start to define a solution: specifically the use of an expert system to represent each decision-maker. Rather than attempt to make any theoretical conclusions as to the workability of this idea, it was decided to follow the universal approach of military simulation groups, in fact most simulationists, and define a prototype. We wished to take a project which had the necessary complexity, used our experience, and, if successful, would provide useful practical results, not just on how to do simulations, but useful in a specific area of military analysis. We chose airborne ASW as the mission area.

THE AIR ASW TESTBED SIMULATION

This Livermore project had its origins in earlier simulations here, specifically in a helicopter-submarine simulation. In order to detect submarines, naval helicopters drop patterns of sonobouys, which listen for the submarine's noise. These patterns can be quite varied, and different sonobuoys can hear and then not hear, as the submarine moves closer to and farther from them. Figure 3 illustrates the variety of signals which might be received in such a process. Coping with this situation, and planning for the improvement of patterns which are having partial successes overtaxed the capabilities of a decision tree arrangement used for this task. It was decided that future simulations may be even more demanding in the area of tactics, and this motivated the new simulation project.

The area of military operations chosen as the testbed for the AI/Simulation fusion experiment was that of land-based aircraft submarine surveillance. In this arena, submarines are initially detected by a variety of sources and sensors, then, sometimes, aircraft are sent to the area of the detection to redetect and further localize the submarine. The topic was chosen for a number of reasons: our experience with sonobuoy simulations would be usable; a naval airbase specializing in these operations was nearby and could serve as a source of expertise; and the requisite degree of complexity was assured [4].

One main source of complexity in this situation is acoustic detection. Underwater sound does not travel in straight lines, owing to the variation of sound velocity with depth, caused by the density increase with depth, and temperature fluctuations caused by mixing and solar heating. Currents also play a very important role, as do the shape and texture of the ocean bottom. In many areas of the world's major oceans, sound from a source near the surface is refracted downward to great depths, and then refracted upward to reach shallow depths again. Thus a listener near the surface could hear a close sound source, then not hear it as he moved further away, then hear it again as he moved yet further. The range of this resumption is from 15 to 40 miles (24 to 64 km.) If the sound source is loud enough, the same pattern repeats itself, with the sound recurring at twice, then three times the original spacing, and so on until absorption and spreading reduce its intensity to below the detectable threshold.

Figure 4 shows how acoustic rays can bend downward from the source, and then bend upwards after having penetrated to great depths.

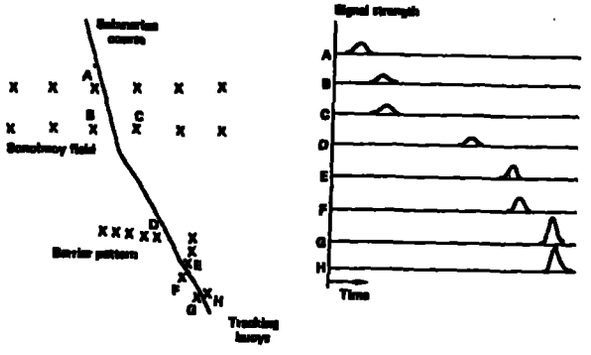


Figure 3: As a submarine traverses a field or a barrier line of sonobuoys, it creates a specific pattern of signals in nearby sonobuoys which must be interpreted to calculate the submarine's track. The track is then used to choose new locations for more sonobuoys, hopefully at tighter spacings to allow more accurate track generation.

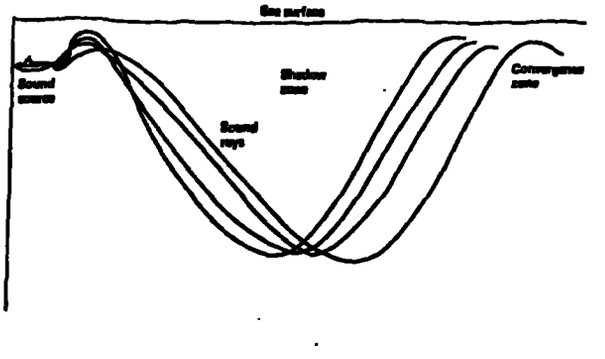


Figure 4: Sound is refracted downward because of the temperature structure near the ocean surface, and upward at deep depths because of the pressure effect on sound speed. The result is a sequence of shadow zones and convergence zones in which a sound source such as a submarine is alternately audible and inaudible.

When an airplane hears noise sounding like a submarine, it must make a series of determinations. Is it actually a submarine? This decision is based on prior detections of the submarine, the presence of ships which might sound like submarines, the time-dependent behavior of the sound, and many other factors. Then the determination of the range can begin. Is the sound from a nearby source, or has sound been picked up after one or more refractions from the source to what is called "convergence zones." This decision is related to the last one. It may be from a distant loud merchant ship, or a close-by submarine, or both of them, interfering with one another.

Besides the complexity of the on-site operations, there is a whole set of complex decisions made even before the aircraft is dispatched. Is there much likelihood of a submarine out there, and just where is it to be searched for? Why not use something else besides an aircraft? Should the search begin now, or wait for one of many reasons—weather, aircraft availability, the hope for more information on the contact, a higher priority mission for the aircraft, and others.

The environment itself provides another source of complications for the decision. Each day the ocean is different, and must be interpreted. Collecting daily data for all the areas of the ocean would be a monumental task, and therefore when a contact occurs, estimates of the ocean condition must be made, unless there is some coincidental data available from nearby.

There is no question that this operational area has enough complexity to require the use of some AI techniques to model the decision-making processes. Actually, as is the case in most military simulations, a critical job in the simulation design is deciding what factors to model, and what factors to approximate and what factors to omit entirely. For the AI/SF project we have a double task, what system features to incorporate in the conventional part of the simulation, and what decision-making to incorporate in the AI portion. These two choices had to be coordinated, and actually never reached a stable set. One situation which arises in the modeling of decision-making by experts is that those modelers doing the work are not experts in that application themselves, and therefore are not necessarily able to predict in advance what are the key features. Only the topic area experts can do this. However, topic area experts are not usually, and in our case not at all, versed in modeling and simulation, so they are not initially capable of advising on what features are key and what are superfluous. It has been an iterative process of key feature identification.

With the project context defined as well as initially possible, we proceeded to design an architecture, first only for the software, as we initially had not conceived of the need for and utility of a hybrid multi-machine hardware configuration.

SIMULATION ARCHITECTURE

The determination of the architecture for the simulation software was a more solvable problem. The first year of the project was spent in determining what modular arrangements were necessary to construct the simulation, and in determining what form the expert system used to represent tactical decision-making should take. While there have been modifications to these structures, and the implementation of the design has led to a relaxation of uniformity of structure, the main features of the design remain as they were initially composed.

The single most important guiding principle used to design the architecture of the software was this: copy the design that military operations follow as much as possible, both in modeling hardware and decision-making. This is a departure from earlier attempts at modeling. Because of constraints on computation, memory size, lack of data, or other reasons, simulations have often had approximations in them which correspond to no physically identifiable phenomena. For example, in Army modeling, because of the difficulty of modeling large numbers of objects, a construct of the "forward edge of the battle area" was used, meaning the line of contact between the two sides. Nowadays, with more powerful computers and voluminous data capacities, individual tanks can be represented in large numbers in simulations. We attempt to adhere to the latter formulation. In decision-making, optimization algorithms have been used to direct fire, move troops, allocate resources, control logistics, deploy defenses, etc. These optimizations are based on very simple models of the activity involved. Nowadays we can include enough complexity in the models of the activities that optimization techniques, such as linear programming, do not apply, and we use instead heuristic rules as the only possible replacement. Our simulation project attempts to install these heuristic rules inside an expert system, as opposed to inside a decision tree or other deterministic structure.

This principle indicates that each physical object be modeled by a block of code devoted to itself. This method has been used before in earlier simulations, but often simulations attempted to crosscut through objects, so that functions determined the modular structure of the simulation code. There would be a module for moving all objects, one for detecting objects, one for communicating, etc. Our current architecture has an object representing a platform. There is a specific identifiable code block for each airplane, submarine, airbase, and other items. This concept is generalized into a total object-oriented structure in the AI/SF code. Each decision maker is an independent object, each communicator is an independent object, each sensor is an independent object, and each communicates with others via messages only.

It would appear that the idea of object-oriented programming originated, possibly separately, in the area of military simulation, before its appearance in the area of artificial intelligence. The motivations for its use in

these two areas are similar but not identical. Simulation is conveniently done in an object-oriented format because this format inherently preserves the natural behavior of objects, the natural partitioning of information between objects, and the requirements for communication as a part of the simulation. Object-oriented languages in artificial intelligence arise more from a desire for a good programming structure, one which is convenient for many applications in AI. It is not necessarily true that object-oriented simulation is more conveniently programmed than a functionally oriented simulation. This may certainly be true in those (unusual) situations where all objects are quite distinct, and require wholly different code for their actions. Usually modern military simulations have multiple units with approximately the same capabilities and activities, which could easily be accommodated in a functionally based simulation.

Functionally based modularization does have a role in an object-oriented simulation. Each object is modularized, within itself, into functionally useful divisions, such as mover, sensor(s), weapon launcher(s), communicator(s), decision maker, and so on, as appropriate for the particular class of object. This second-level modularization is shown in Figure 5, which indicates a separate parcel or page of modules for each existent object.

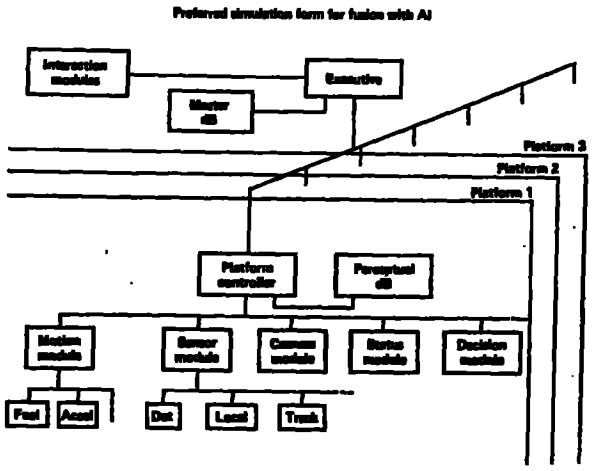


Figure 5: Each object in an object-oriented simulation may have a functionally divided structure, consisting of modules used to move, sense, communicate, make decisions, etc.

Moreover, as noted, this two-level modularization creates an avenue for controlling the information available to each decision maker. In earlier styles of simulation, data about the simulated world was kept in a database somewhere. Special care had to be taken to prevent a decision-making routine from accessing more data than it had a right to. It is important to keep boundaries around the "perceived world view" of each object which makes decisions, and carefully control what crosses these boundaries. This allows us to perform another function which is extremely important in military situations, but which is often ignored in simulations: the corruption of information. Exact information about every object is available in the simulation, and is needed to be able to compute what actually physically happens. For example, exact information on position, course, and speed is used to update the position of each object at time intervals. However, no decision-making object knows the exact position, course or speed of any mobile object, including the one which he may be riding on. He may have navigational sensors which read out his position, but these are inaccurate to a degree which may be important, for example in a rendezvous or coordinated operation, or an intercept or weapon launch. Alternately he may receive messages from another site which is monitoring his position via their sensors. This measurement is possibly inaccurate as well, and may be late in arriving.

More fragile than self-knowledge is knowledge about other objects, on the same side and the other side. Often in earlier simulations, which recognized the importance of information limitation in conflict situations, there were two world views only. One side knew about certain objects, and the other side knew about other objects, not necessarily an identical or a disjoint set. However, with the rise in attention being given to communication modeling over the last decade, it became very clear that not all platforms on one side possessed the same information, and that this influenced the course of events.

Communication between platforms or bases on one side is often difficult or impossible. Sometimes only short-range communication devices are used, because long-range ones give away the location or existence of the transmitting platform. Thus when two platforms physically separate on divergent courses in order to perform separate parts of a mission, or when one platform leaves its home base, it may have its communicated information frozen. All the while it may be adding to its information set by using its own sensors, but knowledge of what other platforms on the same side learn is blocked from it. For example, one plane may learn something about a submarine's location, but it cannot tell its partner over the horizon to change his plan without possibly revealing himself.

We therefore have a multiplication of data-sets in the model. Each object may have some information about each other object, or maybe no information at all. This is a n-squared process, growing as the square of the number of objects, where the true world data base, which contains only the exactly true information about

the position and condition of each object, grows only as n.

Other phenomena vary as n-squared or n also. Communication possibilities vary as n-squared, as any object can theoretically be equipped to talk with any other. Vehicle motion and status changes, such as fuel condition, are n-linear processes. These variations have an important implication. Simulations for naval and air situations typically have many fewer, although more complicated, objects than do land situations. What may be practical for the former may not be practical for the latter. As the number of objects grows, limits in processing time may be reached, and an architecture that is ideal for a small number of objects may be totally unreasonable for a large number of objects. Sampling techniques are often used in land situation modeling to reduce the n-squared processes. Only one object out of a number of them may actually do surveillance in the simulation, although in reality all would have that capability. N-linear processes which occupy a large fraction of the computation time may also be accommodated by sampling. One example is route planning. Only one vehicle out of a group may decide on a route, the lead one. It matters little if each following vehicle follows the exact route of the leader and maintains constant spacing from its immediate predecessor.

Communication possibilities may grow as n-squared, but actual net connections grow only as n times its logarithm. Each object is designated to communicate only with a certain number of subordinates, staff, and superiors, and that number does not grow with the total number of objects. However the number of levels increases as the number of objects does, roughly as a logarithm. The communication hierarchy exists as an initial network, but as a situation progresses, and some objects drop out of the communication net, it must be varied to accommodate the changes and still keep functioning. Thus one set of rules involves who to talk to in what situation, as well as what messages to send.

The communication hierarchy is not the same as the command hierarchy. An object which reports to another certain object may not be responsible for taking orders from it. In other words, the information passing network and the planning network can be distinct. Other rules tell an object whose directions to follow.

PROJECT EVOLUTION

We gained experience initially by building a version of the code to run solely on a VAX, using VMS-Interlisp with an overlay of ROSS (RAND Object-oriented Simulation System) [5]. This proved to operate too slowly, using large amounts of time inside ROSS to pass message traffic in English-like form from one object to another for each event. The next, and current phase, involved adding a pair of Xerox 1108's to the network to perform the AI portions of the computation, while leaving the number-crunching computations to the VAX. LOOPS is the software package on the 1108s [6].

From both a hardware and software point of view, we have a hybrid system for our simulation. The hardware architecture is shown in Figure 6. The VAX runs PASCAL and is responsible for the graphical output of the simulation. It does the world ocean data base manipulation and has routines which perform the function of a human navigator. For example, it examines a map and picks a course from point A to point B in a near-optimal fashion. It can find the nearest point of land, the first point of intersection of a course with land, great circle routes, and other functions. This navigation function is exercised by a navigator object within the AI machines. A typical navigation output for a submarine, as it appears on the graphics output screen, is shown in Figure 7. The VAX also does acoustic computations, and will be responsible for the upgrades of our communications network model.

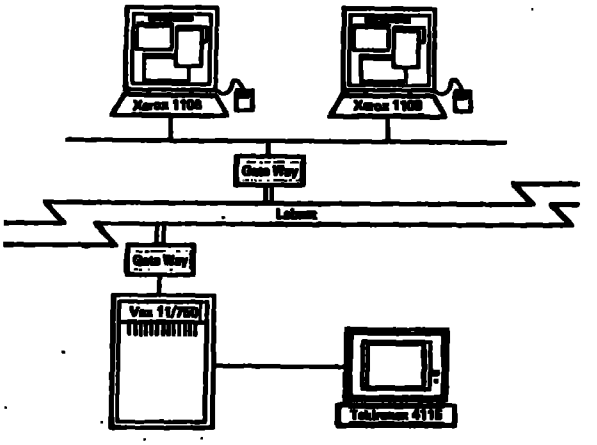


Figure 6: A hybrid computer system for AI/Simulation work can consist of one or more LISP machines, coupled to a standard mainframe for more intensive numerical computations. Graphics output is typically one of the most demanding activities of the simulation and benefits from a tight coupling to the mainframe. The version shown here is that currently operational at LLNL.

The same philosophy is designated for all databases in the simulation. The data is separated out from the remainder of the simulation, and is accessed via specialized routines. These routines and the database constitute an object in the object-oriented framework. This process ensures that the databases will not be altered inadvertently.

These objects, the database manipulators, and several other objects are referred to as servant objects or shadow objects, to distinguish them from objects which have a corresponding entity in the real world which is being simulated. The "timekeeper" object might be considered most

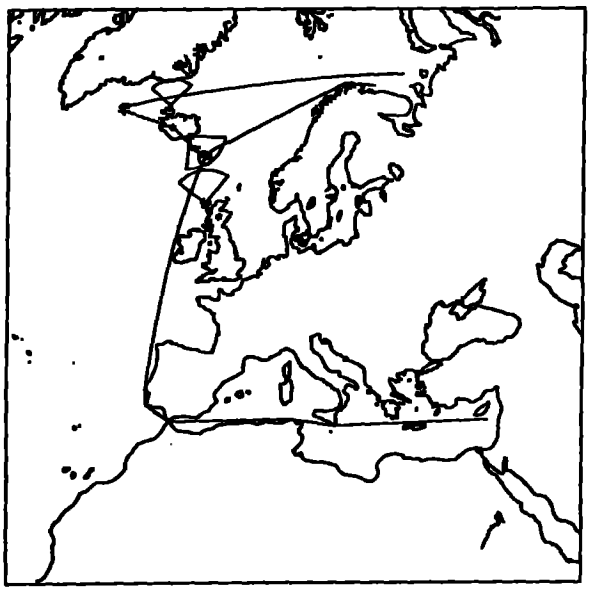


Figure 7: The navigator object serves to locate paths through the world oceans for ships and submarines in the AI/SF simulation.

important, in that it controls the progress of the simulation. All objects have the same level; there is no hierarchy of objects, except for inheritance. We allow objects to inherit properties from generic objects. For example, each aircraft of any sort inherits properties from the generic aircraft and is an instance of it. This is a programming convenience, and is facilitated in ROSS as well as LOOPS. An aircraft inherits its motion module, and has its own parameters inserted into it, such as fuel level and consumption rate.

For each new part of the code, we have to make the decision as to whether to put it into the VAX or into the LISP machines. So far there have been no ambiguous cases.

We now have a running code, with about 10 separate decision-making entities, each having up to 40 rules. Our decision has been to keep the individuals relatively simple for one reason. There are a myriad of other questions of architecture, module design, interobject communication we need to answer, and to attempt to build very sophisticated rulesets for individual actors would use up too much effort, and interfere with other investigations.

Three areas where we feel we have learned from the design and implementation work are in the AI module design, in the human-to-human process of developing the tactical knowledge base in the context of multiple intensely interacting decision-making entities, and in the use of interactivity to improve programming and verification ease. These three are discussed in sequence.

AI MODULE DESIGN

Besides structuring the code in a global fashion, it has been necessary to structure the modules themselves. Fortunately, there is much experience available for such things as modeling detection, motion, tracking, sensor integration, etc. This is not so for the AI portions of the task. It was necessary to come up with a rule set structure, as well as a way to keep the AI portions of the code manageable.

Rule-based systems tend to use large amounts of time, as they grind through their rules checking them for applicability. To solve this problem we have instituted two arrangements. The first is the hierarchy of rulesets. Instead of checking all rules, we packed them into separate blocks, and do not invoke two blocks simultaneously, nor serially unless one calls another one.

It would be best to imitate the natural structure of rules that tactical experts have in order to implement our expert system. However, in our elicitation of rules, we find that a natural hierarchy does not exist, is not used, or is not apparent to most decision makers. Instead, it was necessary to impose one in order to bring order to the rules of various experts. Military science indicates that one starts a military problem by deciding on a mission, after examining the situation in detail. Each mission is then decomposed into tasks, which divide into operations.

There is a discordance between what might constitute a mission on one command level and on a different level. The basic purpose is clear, however. There should be a derivation of successively more complicated directions for actions from more global directions. To imitate this we build a hierarchy of blocks of tactical rules. This arrangement of ruleset blocks is shown schematically in Figure 8.

This however is not the principal means of controlling the time usage by the AI modules. We create for each decision-making object a separate block of rules called an agenda. Essentially the agenda serves as a filter for its object. Rules in the agenda are of the form:

If (condition A) then (invoke block X of object N)

One of the outputs of the rulesets is to add rules to or remove rules from the agenda belonging to its object. The agenda serves to cause an object to scan its ruleset, rather than have its scanning initiated by the clock at each timestep.

The idea is not complicated. Out of each object's decision-making ruleset we withdraw a small subset of rules which contain key decisions on when to make a change in plans or procedures. An example might be to reevaluate a mission after receiving a message from a superior in the command chain, or, more specifically, to break off laying a pattern of sonobuoys to listen for submarines if the first half of the pattern has already gone "hot" and has a detection.

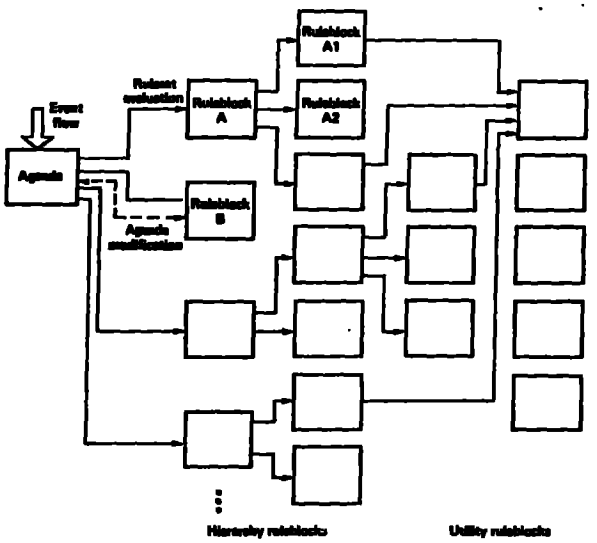


Figure 8: Ruleset blocks naturally fall into two groupings, one hierarchically connected, starting with the agenda, and the other a set of utility rulesets, such as one to determine the nearest point of land--which is used in several ruleset blocks.

Many of these agenda items concern available data, and are the result of temporizing decisions. When a decision-making module is invoked to replan, it may decide to do any of the set of actions it is capable of. Alternately it may decide to go into a wait state, in the anticipation of better information arriving temporarily. This will result in an item being installed in its agenda, which says, when a message from object M arrives [with information I], reinvoke the ruleset. Alternately and more frequently the agenda item will simply read, wake me up at x o'clock, or more exactly

If (time > t_0) then (invoke ruleset Z of object N).

Then the ruleset looks to see if it has gotten more data, and if not, it may decide to act anyway. This form prevents permanent sleeping of an object which has a decision to make.

The agenda is such an important item for making the simulation feasible that it occupies a major part in the architecture of the simulation. As shown in the architecture diagram of Figure 9, the agenda serves to monitor the perceived database of the object it belongs to. This perceived database is continually updated with corrupted information via sensor modules from the world database, the repository of true knowledge about objects in the simulation. When an agenda rule correctly fires, it kicks off a ruleset block in the object which owns it. One possible result is for the object's total ruleset base to modify the agenda so that essentially the object is now looking for something else important, based on what has just happened.

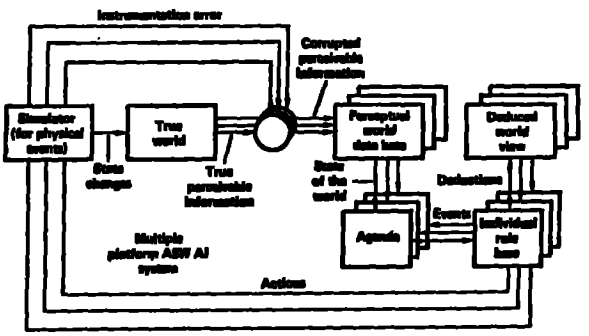


Figure 9: The architecture diagram shown here emphasizes the internal arrangement of code modules relating to the AI portion of the simulation.

TACTICAL KNOWLEDGE BASE DEVELOPMENT

A number of interesting questions arise during the process of eliciting expert knowledge from tactical/allocation experts. Many decisions are coordinated decisions. By this we mean that two or three command levels have long ago agreed on what procedures will be followed in what situations, and what constraints will be applied. This happens in situations which are frequently exercised. It also happens for another less obvious reason. Military officers move up the hierarchy of ranks, and often take over command of their previous post, having trained a deputy to follow rules that they are accustomed to. Thus the higher command level has an expert who formerly was the expert at a lower level, and who trained his replacement to use the same procedures as he did. In such instances coordination is marvelously simple.

However, from an expert system point of view, finding out who actually makes a decision is not so easy. For procedure-following situations, it matters little what level makes a decision and what level agrees, does not overrule, or advises. However in the less frequent but much more critical cases of unusual situations, there is no precedent for who is to make exactly what decision, and the expert system implementer is the one who makes the decision. Possibly this reflects the fact that in an unusual situation, the real decision maker is the one with the strongest personality or most clear conception of the situation, or some other aspect, rather that a designated command level.

Another intriguing question arises from the multiplicity of objects at the same level. If there are a hundred planes flying at once in a major tactical situation, there are a hundred experts being invoked. Should they all be copies of the same expert? If so, we run the risk of

relying on a less-than-best expert system, and basing system design decisions on imperfect tactics. The other side of the question says that it would be good to have a variety of expert systems doing the same job, so that the actual existing variety of tactics would be employed. This means more work for the elicitation group, who must now talk to more than one expert for each type of platform. One way to partially satisfy the desire for an assemblage of experts rather that a set of clones is to allow parameter variations, rather than more substantial ruleset variations. For example, one expert may respond to a contact based on a signal just over the threshold of detection, while another may wait until a stronger signal is obtained. One expert may wait for an hour for better data, while another may wait two. This randomization is not the only one in the model, because of the necessary randomization in detection processes, and may not be even visible in the results.

Planning in the AI/SF project shares the same problems as other AI planning systems. We have a list of mission requirements, and try to satisfy the first one with a plan, such as a course of flight. Then the second mission requirement is checked to see if it is satisfied automatically by the chosen flight plan. If so, the next one is checked. If not, however, a variation is tried which also satisfies the first mission, and which may or may not satisfy the second. If several tries do not satisfy the second mission, it is ignored and the third one examined. This continues until the mission list is exhausted. While this procedure does not have a grain of intuition, it can produce an acceptable plan in that it must satisfy the most important mission to be considered.

INTERACTIVITY

There are several reasons for building interactivity into the simulation framework. One is certainly ease in debugging. Many programming errors have been first determined by noticing that an object is moving to a nonsensical place, such as an aircraft searching for submarines under the Greenland icecap, or landing on a port rather than an airfield. It is much easier for the eye to pick out which object is errant, than to evaluate final destinations or any other listed data. But there are more ways to utilize interactivity for code improvement and testing than by just observing the operations.

Each object has a view of the world, his "perceived world view." He makes his decisions on what actions to take based on this "PWV," rather than on what actually happens in the simulation's true world. By simultaneously displaying what the true world looks like, in terms of object positions and motions, and what an object's "PWV" looks like, a discrepancy can be observed, as it happens. For example, by showing the actual position of a submarine, and an airplane's idea of where the submarine is, and watching them diverge, we can see how the aircraft loses its target. This may lead to rapidly seeing where a code error is. If developed more, it may be a powerful tool for improving tactics. In Figure 10 we show how the architecture has to be modified to take advantage of the comparisons between the "PWV" and the true world.

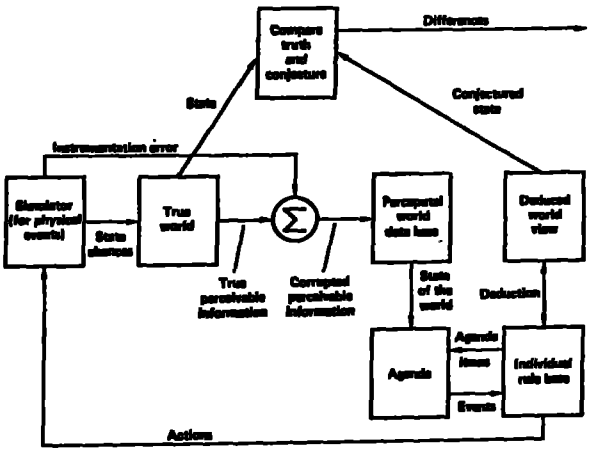


Figure 10: Debugging the rulesets is aided by monitoring the deduced world picture, for example where a pilot thinks a target submarine is, and comparing it with the true world view, i.e., the actual submarine location. A sudden divergence of the two signals is a ruleset problem.

A second way of using simultaneous views to understand problems in the code, is to show the "PWV" of an object twice, once from a normal run, and once from a run in which the errors have been zeroed out. Then if an airplane fails to complete its observations, it is immediately obvious if it is caused by too inaccurate sensors, making too large errors in the received data.

One problem with completely automated simulations, that exists to a lesser extent with interactive simulations, is the set-up time. Before a simulation can be run, objects have to be emplaced, given plans, and many parameters set. This can be done interactively, with a graphics screen. If the software is properly created, plans for search paths and buoy patterns can be laid down. The use of menus can be used for parameter setting, at some possible savings of time.

CONCLUSIONS

We have learned that a hybrid architecture for military simulations can be made to work effectively, and that it is superior to use of a single machine. Programming ease is improved by having hardware tailored to number crunching and symbolic manipulation, both of which are required in typical military simulations.

We have learned that an object-oriented format arising from simulation requirements mates well with the object-oriented formalism of AI. In fact, it is not clear how any other structure for the simulation would permit this fusion.

We have realized that the use of an agenda facilitates expert system interaction with the rest of the system, and that graphical output is useful for the final stages of debugging even the AI portions.

We have discovered a new problem area in the elicitation of expert system knowledge, the difficulty of establishing command authority for each specific decision in a multi-level chain of command. We have developed a modus operandi of dealing with the ambiguity.

We have learned that geographic database control is important to include in the initial design stage. Military affairs draw much of their complexity from the environment (geography, oceanography, weather) and this database is called on extensively by the expert systems representing tactical officers. We have found it convenient to build interface modules to handle inquiries to each database, and not to allow any direct contact between AI modules and databases. In a sense, we have built oceanography experts, for example, to answer the questions of a CO about ocean conditions.

We have learned the importance of dividing up military combat units into several objects, including at least a commanding officer, a communicator, and a performer of operations. Trying to accomplish all functions within a single database imposes unneeded complexity. We also have appreciated the fact that appropriate message delays need to be built into the simulation from the outset. These delays are critical in running a multiple machine configuration effectively.

We have learned that English-like syntax is not feasible on the machine we now have, and that direct interaction of military officers with the simulation needs to be mediated by LISP experts familiar with the structure of the AI modules.

We have learned the utility of involving topic area experts in the initial planning cycle for a simulation, for the purpose of defining key features.

At this point we continue to evolve the simulation toward a more impressive demonstration of the ability of AI to be useful in military simulation, and to face problems caused by strains on computation resources, both in decision-making and database manipulation.

References

1. Huber, Reiner K., ed., Systems Analysis and Modeling in Defense, Plenum, New York, 1984.

2. Huber, Reiner K., et al., eds., Military Strategy and Tactics-Computer Modeling of Land War Problems, Plenum, New York, 1975.

3. Hughes, Wayne P., Jr., ed., Military Modeling, Military Operations Research Society, Washington, D.C., 1984.

4. Erickson, Stanley A., "The AI/Simulation Fusion Project at Lawrence Livermore National Laboratory," UCRL-90777, Livermore, CA, 1984.

5. McArthur, Davidand Philip Klahr, "The ROSS Language Manual," N-1854-AF, Santa Monica, CA, 1982.

6. Bobrow, Daniel B. et al., The LOOPS Manual Extended Documentation, Xerox Corporation, Sunnyvale, CA 1985.